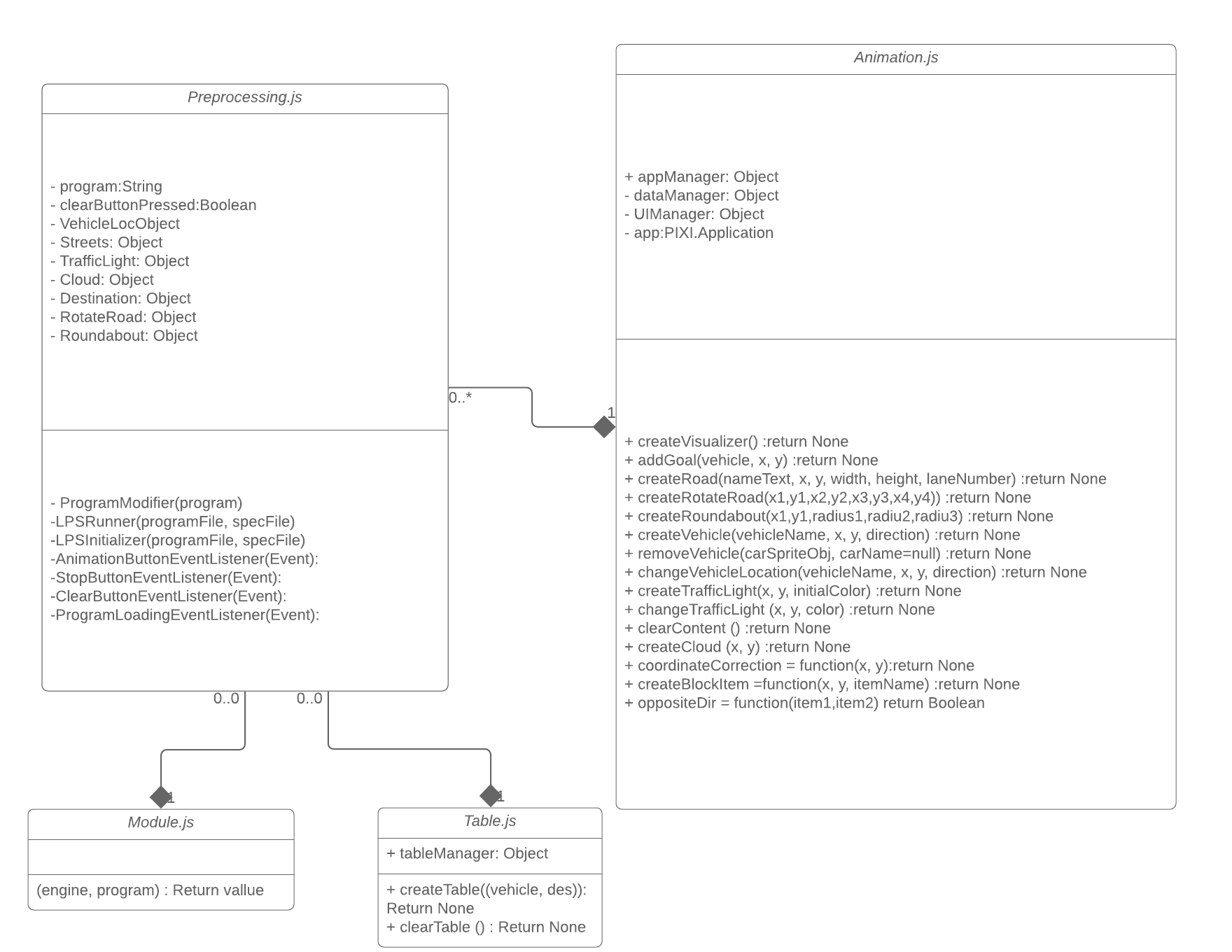
Front-end logic design and animation

This chapter I will introduce the underlying front-end side of LPS visualiser. Which include three main part. Part one is the animation part which is mainly about the PIXI canvas. The second part is the link between the animation and data (fluent/facts) how is the data parsed. The third part is about the layout design using twitter bootstrap and other open source toolkit.



The class diagram of the front-end side is structured as above. The

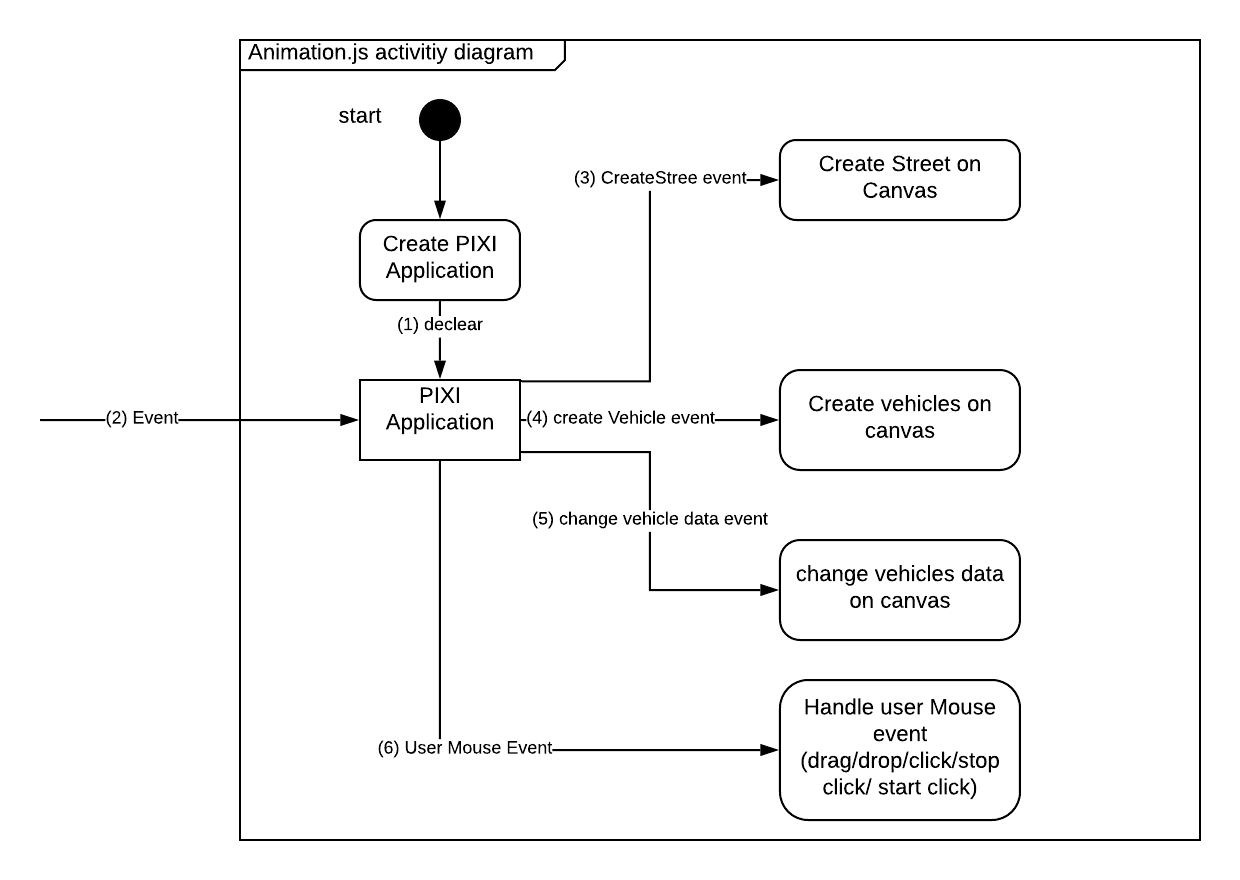
3.4.1 Animation.js

3.4.1.1 introduction

The Animation.js create the PIXI canvas application for animation and read the changing fluent in LPS program simultaneously. To achieve this, a concurrency model need to be used. Animation.js use a range of data structure such as AppManager to matain different list of object, UIManager to maintain a range of UI component.

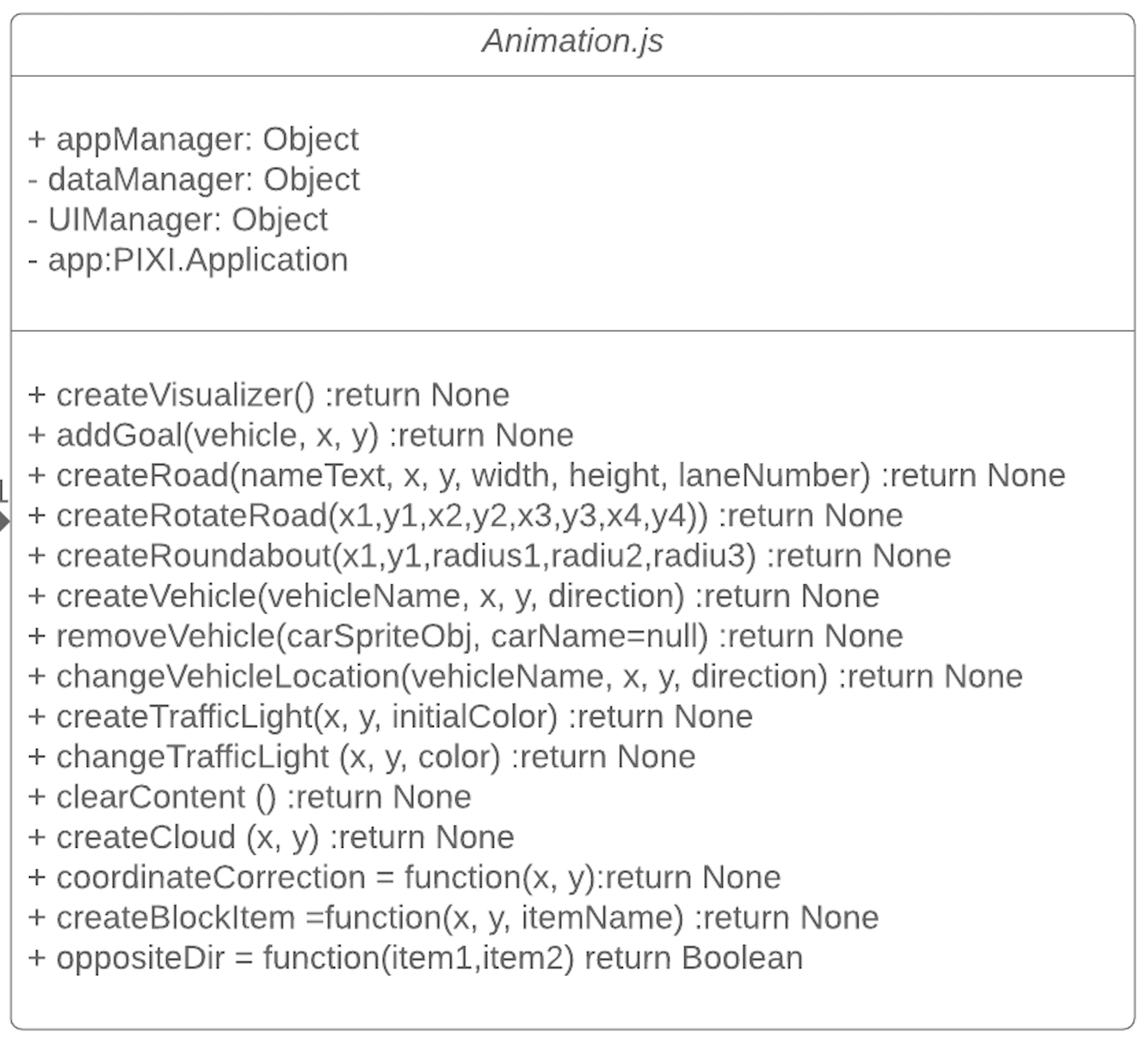
3.4.1.2 structure

The structure of the animation is showed as an activity flow chart. The animation.js will run once the user is on the animation page. Due to the DOM structure introduced in Section? The PIXI application will be created as soon as the page is loaded. The different evnet happeded will fire different function. (1) is the creation of the PIXI applition. PIXI has a tree data structure every component play a child node role in the whole application whenever a component is created it need to be append into the application tree.



1. Describe about the creation of the PIXI application
2. Is different event whether is the event of the changing state of fluent or the user mouse event
3. Handles the street creation. When street predication as a fact in LPS program is read into the LPS interpreter.
4. Handles the creation of the vehicle When vehicle predication as a fluent in LPS program is read into the LPS interpreter.
5. Handles the information changing of the vehicles, including direction, location, speed while the runtime of lps program. State of location predication will be detected by the engine and the PIXI application handles it by change the location on canvas accordingly.
6. Handles the user event while includes the clicking event (click on car to change direction, click the stop button, click the animation button). Drag event, drop event(drop in deletion, drop on street). Some globel information might change depends on the action that user performs.

3.4.1.3 class fields



There are four key components for Animation.js

Firstly, the appManager maintain the fluent entered by user such as vehicle, street, light, blocked item and so on.

DataManager maintain the response used to detect user whether drop and drop the vehicle or just click on the vehicle to change the direction. Car counter is used here as an Integer type also used here

3.4.1.4 functionality

createVisualizer Initialise the canvas, all the buttons, title text, define the base graphic

addGoal = function(vehicle, x, y) adding goal tag onto the map

createRotateRoad = function(x1,y1,x2,y2,x3,y3,x4,y4) This field will create the rotated road depends on the user defined

reateRoundabout = function(x1,y1,radius1,radiu2,radiu3) This field will create the roundabout

createVehicle = function (vehicleName, x, y, direction) This field will create the vehicles with name, location and direction.

removeVehicle = function(carSpriteObj, carName=null) Will remove the vehicle if user drop the vehicel in the deletion area

changeVehicleLocation = function (vehicleName, x, y, direction) this field will modify the child in app

createTrafficLight = function(x, y, initialColor) this field will create the traffic light based on the location

changeTrafficLight = function(x, y, color) this field will change the traffic

clearContent = function ()this method will clear out the street and vehicle information

\* the basic canvas will remain the same (coordinate axis and the title text)

createCloud = function(x, y) this method will create the cloud and animate the cloud in the canvas

coordinateCorrection = function(x, y) correct the coordinate on to the street. use might place the car onto other part rather than the road

createBlockItem =function(x, y, itemName) When there is a blocked item placed by user/program the front end will show up a cross

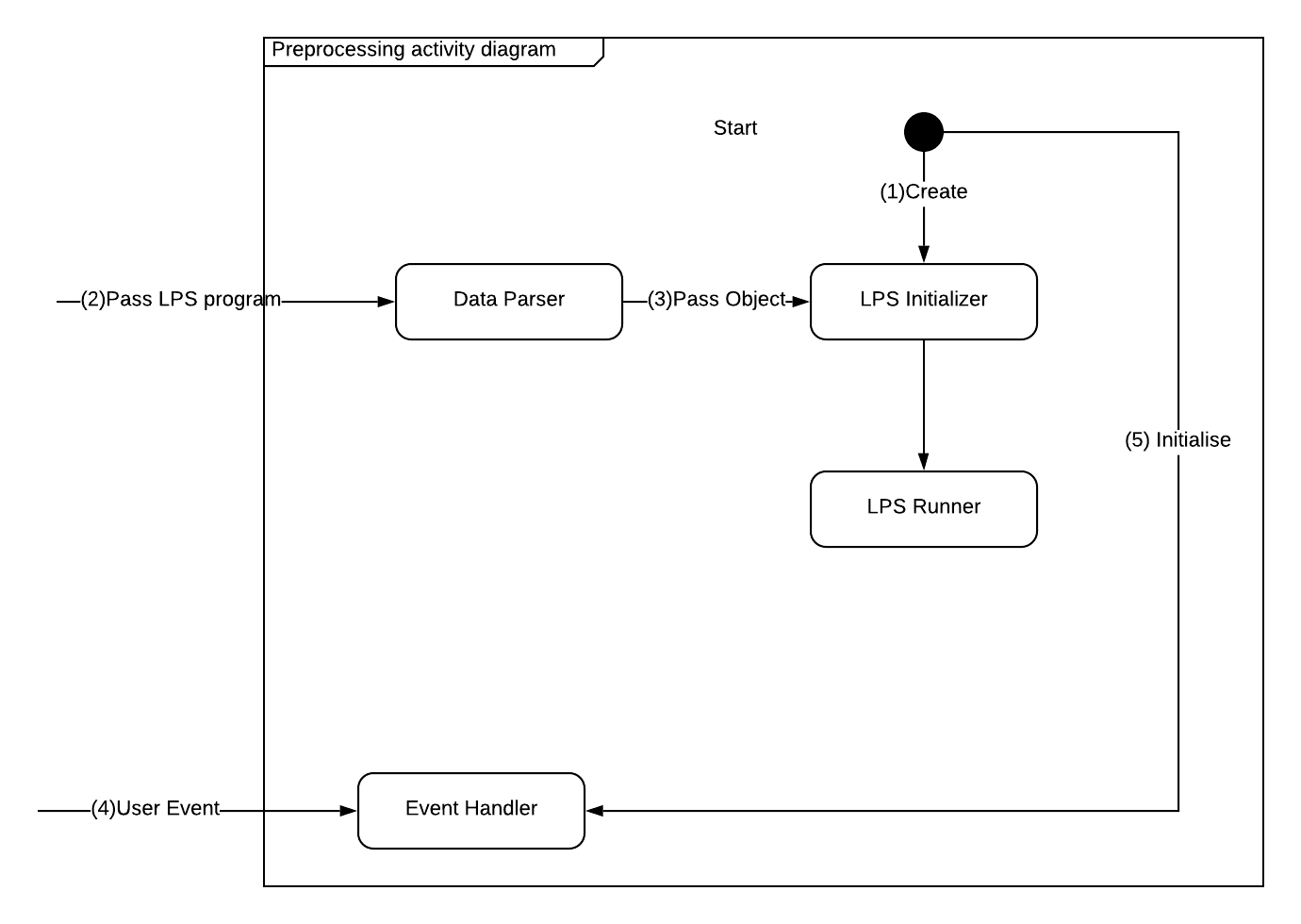
oppositeDir = function(item1,item2) This part is checking all the adding cars that is there is any car that is opposite to each other (same horizontal or vertical level)

3.4.2 preprocessing.js

3.4.2.1 introduction

The preprocessing.js handls the user input (except the input on PIXI canvas) such as mouse click the animation button, load the LPS program. Also preprocessing parse the useful information passed into PIXI app, such as vehicle location, street inforaton. To achieve this, Document Object Model is introduced here.

3.4.2.2 structure



The diagram shows the activity diagram of the preprocessing.js.

(1)The web page will create a LPS initializer when user passed LPS program into the class.

(2)(3)The passed LPS program is stored as string value so it need to be parsed into different object. The data which is parsed including the different road traffic light, vehicles and so on. Once the LPS program is passed in and prased. LPS initializer will call the animation class to create canvas vehicle etc.

(4) The preprocessing.js class also handels the user input using Document Object Model of JavaScript. Such as user load the LPS program, user type in the text area. In Event handler a program modifier is also needed incase the user has pre modified the program before running the intreperter. For esample, user loaded the LPS program and user change the direction of the vehicle via PIXI canvas (LPS program has not been modified), the program modifier need to catch that modification such that the canvas has same information as LPS program.

3.4.2.3 Document Object Model (DOM)

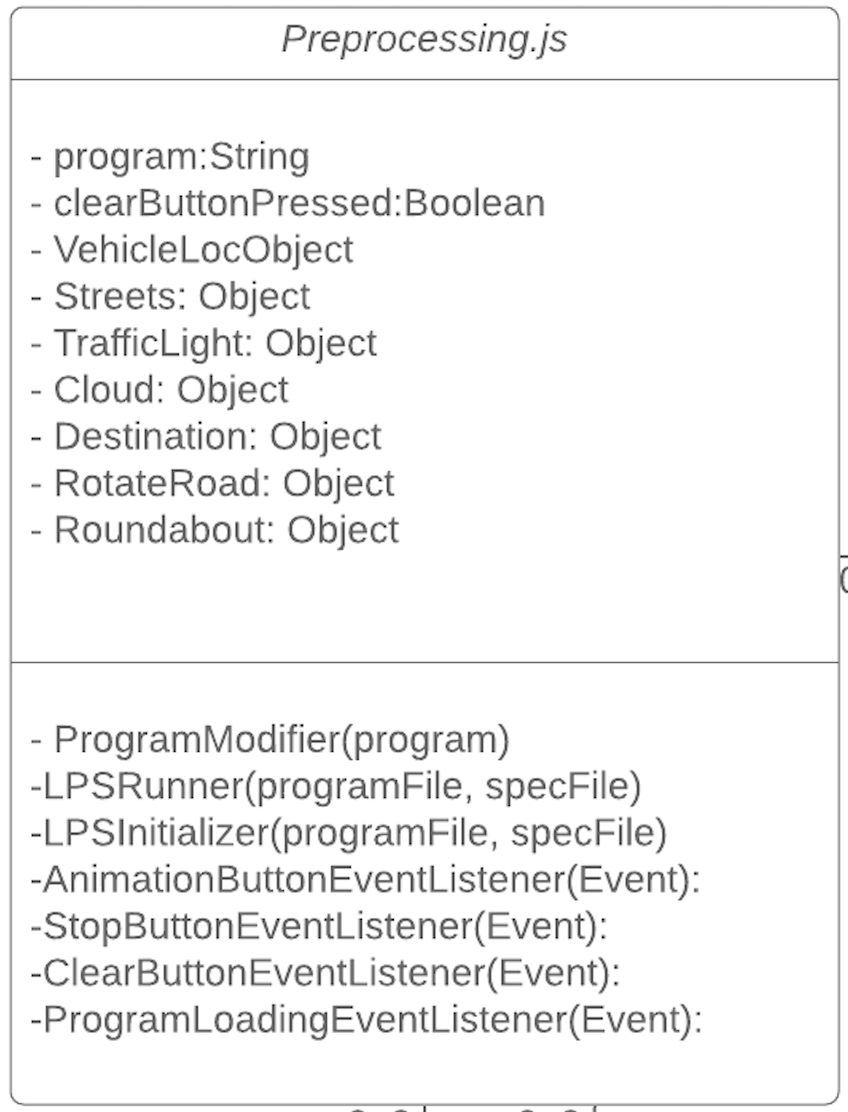
The Document Object Model is a programming API for documents. The object model itself closely resembles the structure of the documents it models. For instance, consider this table, taken from an HTML document: the change on the html can be easily detacted and change via javascript and handle event accordingly. For example, when user load the LPS program this event will be handed and the LPS program will be parsed as string text into a text box area.

By define javascript class as the follow structure we can also avoid the ordering issue [<http://xahlee.info/js/js_executing_order.html>]

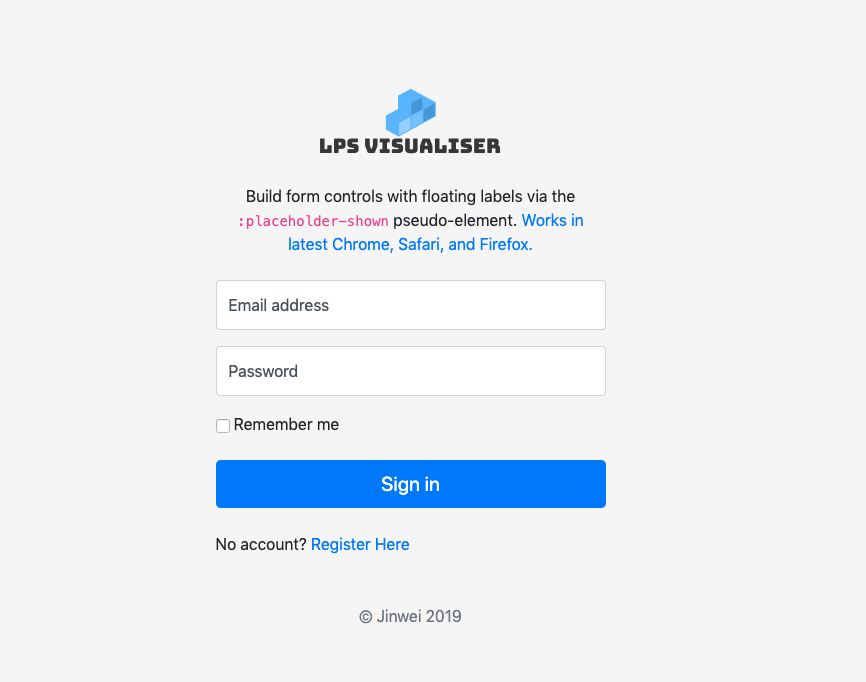
(function (window) {

})(window);

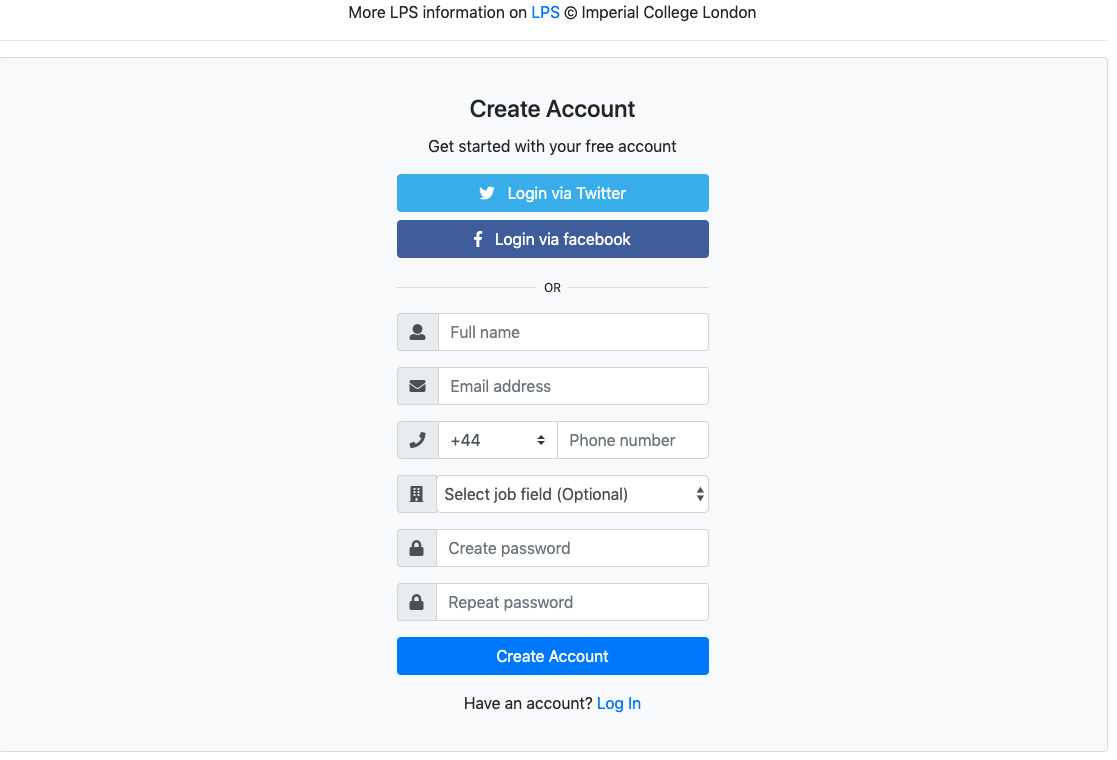
3.4.2.4 functionality



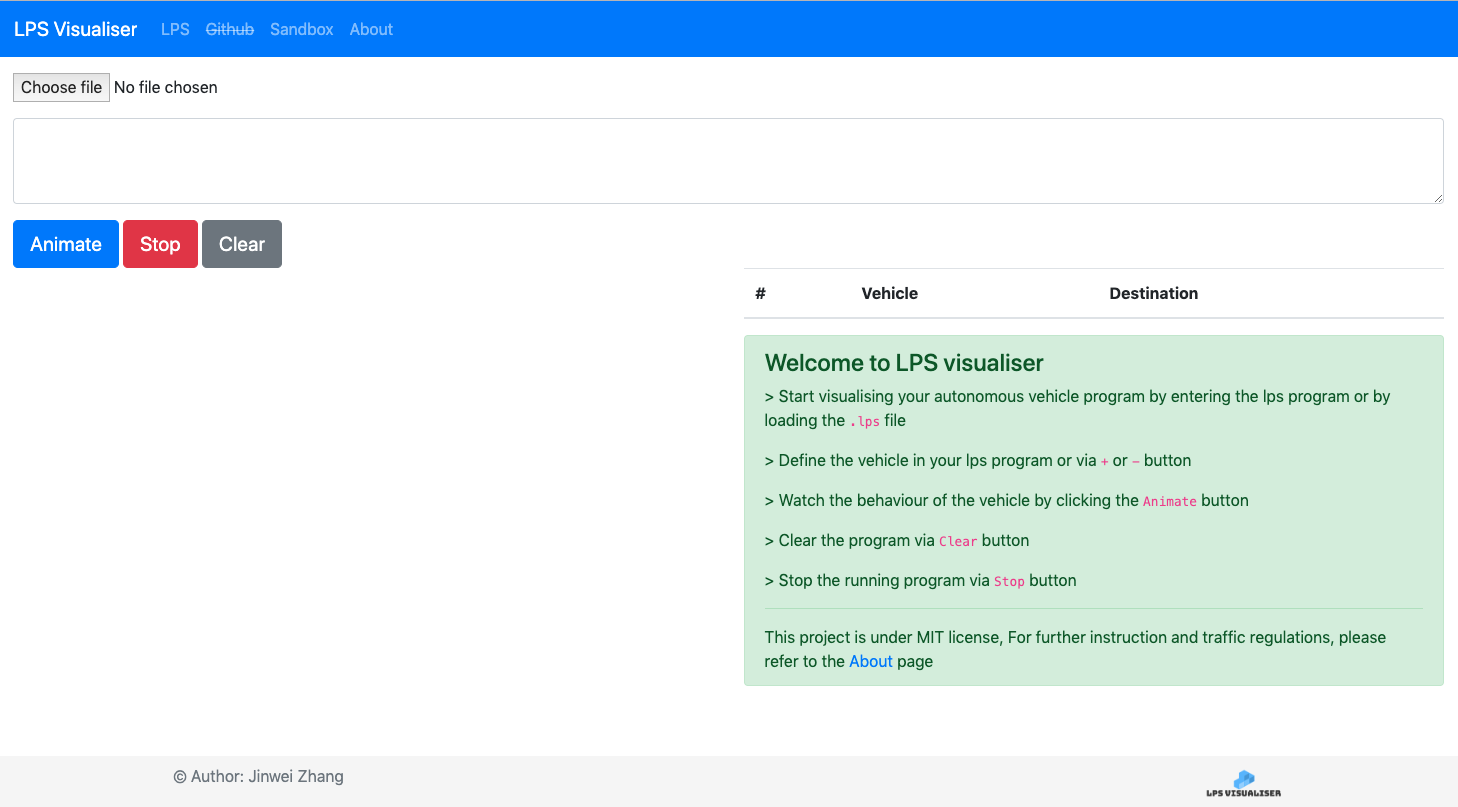
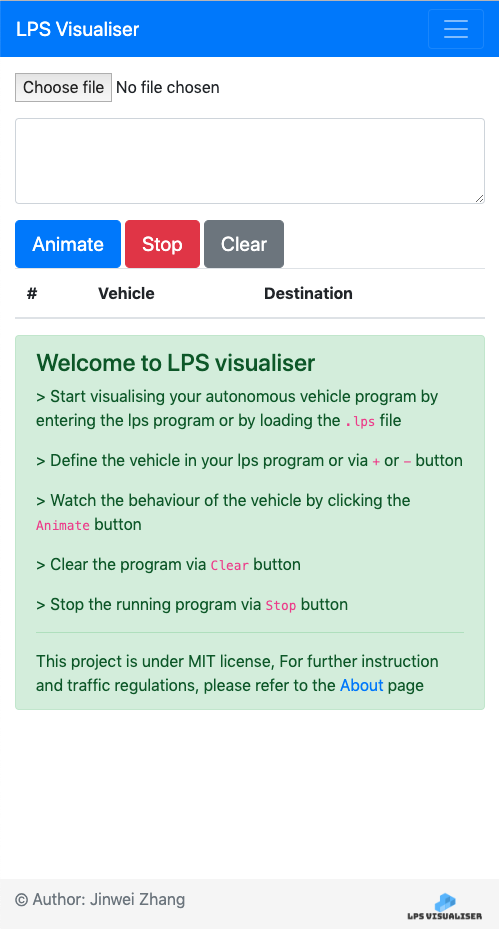
3.4.3 Demonstration Website design



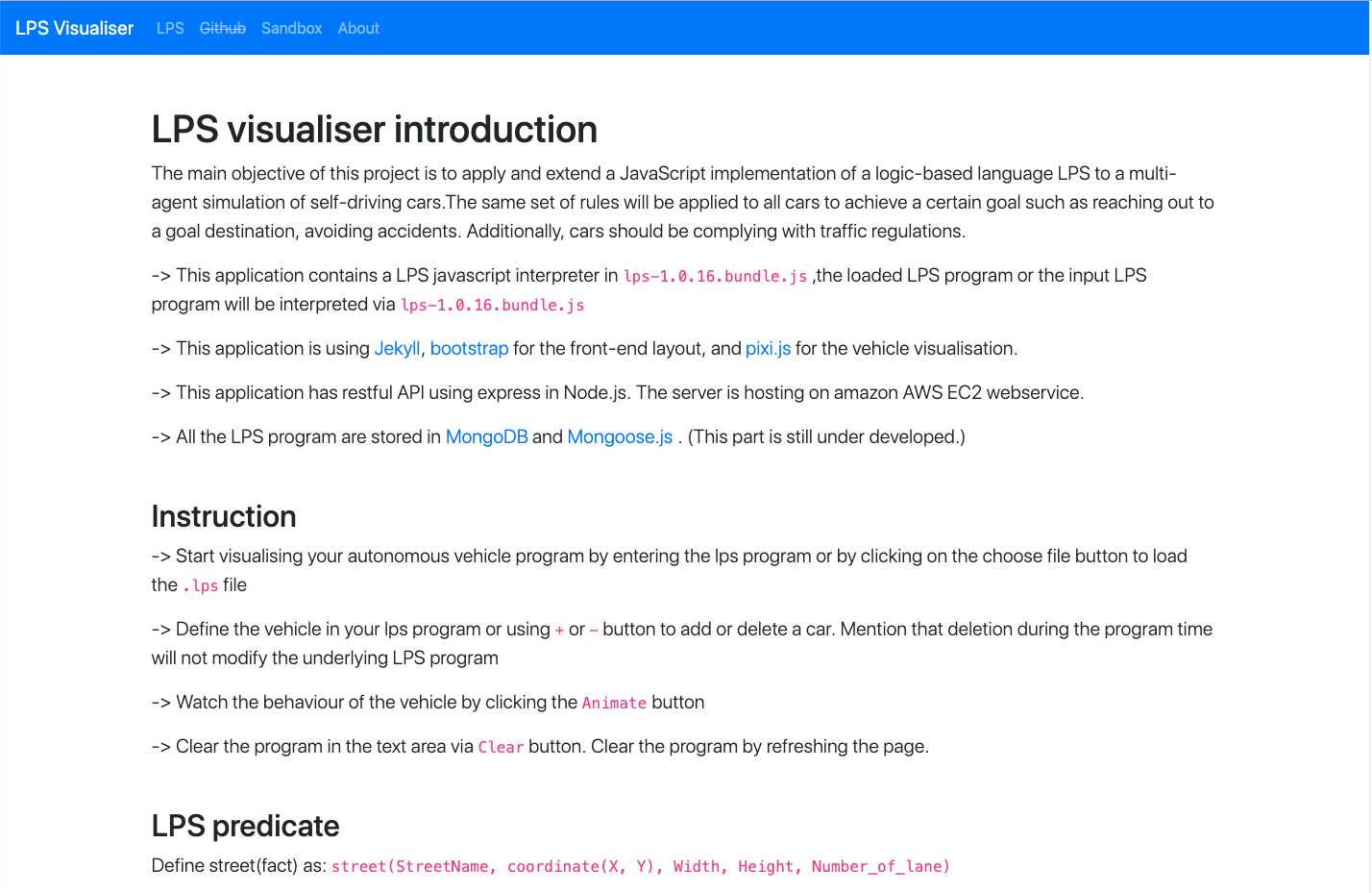
the login page



the registeration page

the main page by using boostrap it is also mobile friendly.



about us page. This page give a brief introduction of how to run the program and the technology involved.

3.4.3.1 structure

3.4.3.2 API and open source toolkit

boostrap

Jekyll is used in login page and register page.

Font Awesome icons and css are used in register page.

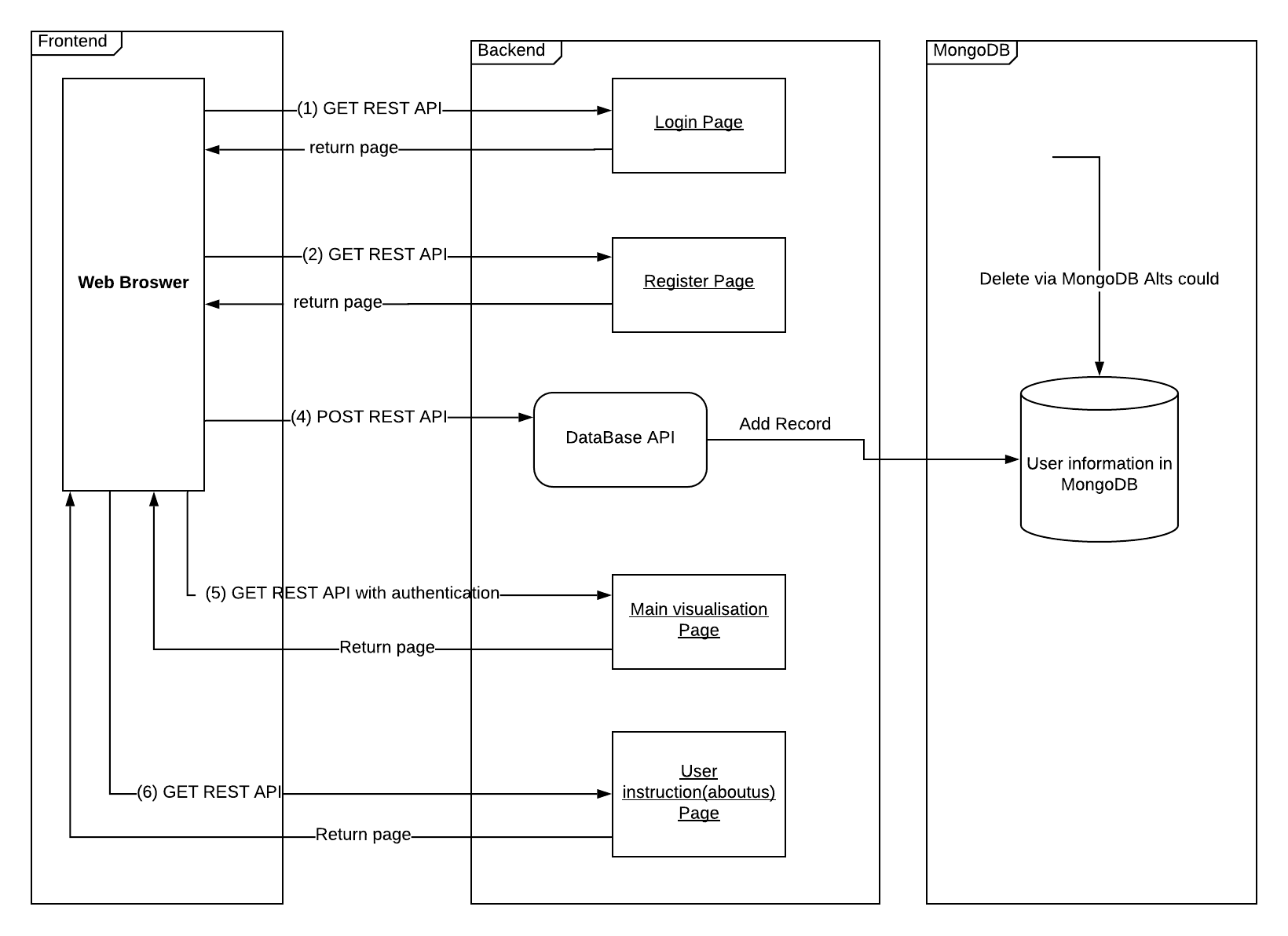
3.5 back-end hosting

This chapter will introduce about the back-end of developing. The reason why backend is introduced here is because website can be host and be accessed by everyone. Hence more function can be introduced here such as host LPS program in database, addition of LPS program into data base via user interface etc.

3.5.1.1 introduction

The back-end of the project is host on AWS server using express framework.

3.5.1.2 structure



3.5.1.3 Express.js

3.5.1.4 MongoDB and mongoose

const UserSchema = new mongoose.Schema({

name: {

type: String,

required: true

},

email: {

type: String,

required: true

},

phone: {

type: Number,

required: true

},

job: {

type: String,

required: false

},

password: {

type: String,

required: true

},

date: {

type: Date,

default: Date.now

}

});

3.5.1.5 Passport.js

3.5.1.6 AWS EC2 server